Отчет лабораторная 4. Вариант 34

Задание 1. Установка sqlite в macOS

Установил SqLite:

```
alex@AVOCADOBOOK:~$ sqlite3
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite>
```

Задание 2. Управление базой данных из консоли

Упражнения 2.1-2.3 выполнил

создание таблицы (create);

```
sqlite> .tables
sqlite> CREATE TABLE product (
   ...> id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
   ...> name TEXT NOT NULL,
   ...> price INTEGER NOT NULL,
   ...> days_left INTEGER,
   ...> sort TEXT NOT NULL,
   ...> date INTEGER NOT NULL,
   ...> store_date INTEGET NOT NULL );
sqlite> .tables
product
sqlite>
```

вставка данных в таблицу (insert):

```
sqlite> INSERT INTO product ( name, price, days_left, sort, date, store_date )
   ...> VALUES ( 'fish', 55, 4, 'red', strftime('%s', 'now'),  strftime('%Y-%m-%
d','2022-03-29') );
sqlite>
```

```
sqlite> INSERT INTO product ( name, price, days_left, sort, date, store_date )
   ...> VALUES ( 'bred', 49, 7, 'white', strftime('%Y-%m-%d', '2022-03-20'),  st
rftime('%Y-%m-%d','2022-03-27') );
sqlite> INSERT INTO product ( name, price, days_left, sort, date, store_date )
   ...> VALUES ( 'meat', 70, 5, 'beef', strftime('%Y-%m-%d', '2022-02-15'),  str
ftime('%Y-%m-%d','2022-02-20') );
sqlite> INSERT INTO product ( name, price, days_left, sort, date, store_date )
   ...> VALUES ( 'popit', 4, 100, 'colorful', strftime('%Y-%m-%d', '2021-11-14')
,  strftime('%Y-%m-%d','2022-01-20') );
sqlite> INSERT INTO product ( name, price, days_left, sort, date, store_date )
   ...> VALUES ( 'cheese', 59, 20, 'bulgarian', strftime('%Y-%m-%d', '2022-02-25
'),  strftime('%Y-%m-%d','2022-03-15') );
sqlite> INSERT INTO product ( name, price, days_left, sort, date, store_date )
   ...> VALUES ( 'silk', 150, 150, 'chineese', strftime('%Y-%m-%d', '2020-07-24'
),  strftime('%Y-%m-%d','2022-03-10') );
sqlite>
```

выборка данных (select) с выводом всех данных по столбцам и строкам, с сортировкой по id и по имени и с выводом последних 5 строк (инструкция limit);

```
sqlite> SELECT *
   ...> FROM product
   ...> ORDER BY id DESC
   ...> LIMIT 5;
6|silk|150|150|chineese|2020-07-24|2022-03-10
5|cheese|59|20|bulgarian|2022-02-25|2022-03-15
4|popit|4|100|colorful|2021-11-14|2022-01-20
3|meat|70|5|beef|2022-02-15|2022-02-20
2|bred|49|7|white|2022-03-20|2022-03-27
sqlite>
```

выборка данных с фильтрацией (условие where), если id=5;

```
sqlite> SELECT *
   ...> FROM product
   ...> WHERE id = 5;
5|cheese|59|20|bulgarian|2022-02-25|2022-03-15
sqlite>
```

выборка данных с фильтрацией (условие where) и с совпадением по маске, например все записи, где имя объекта (согласно варианту) начинается на первую букву вашей фамилии (инструкция like);

```
sqlite> SELECT *
   ...> FROM product
   ...> WHERE name LIKE 'c%';
5|cheese|59|20|bulgarian|2022-02-25|2022-03-15
sqlite>
```

переименование таблицы (alter);

```
sqlite> ALTER TABLE product
   ...> RENAME TO products;
sqlite>
```

обновление данных с использованием update;

```
sqlite> UPDATE products
   ...> SET price = 199
   ...> WHERE name = 'silk';
sqlite>
```

удаление строк по id и по названию объекта;

```
sqlite> DELETE FROM products
   ...> WHERE id = 2;
sqlite> DELETE FROM products
   ...> WHERE name = 'cheese';
sqlite>
```

Результат выполнения предыдущих команд:

```
sqlite> SELECT *
   ...> FROM products;
1|fish|55|4|red|1648242413|2022-03-29
3|meat|70|5|beef|2022-02-15|2022-02-20
4|popit|4|100|colorful|2021-11-14|2022-01-20
6|silk|199|150|chineese|2020-07-24|2022-03-10
sqlite>
```

экспорт базы данных в файлы .sql, .csv.;

```
sqlite> .header on
sqlite> .mode csv
sqlite> .output Products.csv
sqlite> SELECT * FROM products
   ...> ;
sqlite> SELECT * FROM products;
sqlite> .quit
alex@AVOCADOBOOK:~/Desktop/study/4 sem/TP/lab4$ ls
Products.csv
alex@AVOCADOBOOK:~/Desktop/study/4 sem/TP/lab4$
```

```
sqlite> .once Products.sql
sqlite> .dump
sqlite> .exit
alex@AVOCADOBOOK:~/Desktop/study/4 sem/TP/lab4$ ls
Products.csv  Products.sql
alex@AVOCADOBOOK:~/Desktop/study/4 sem/TP/lab4$
```

удаление таблицы;

```
sqlite> DROP TABLE products;
```

```
sqlite> .tables
sqlite>
```

Упражнение 2.4. Выполнить запросы по вариантам

Выполнить запросы:

• Вывести данные про товары срок годности которых истекает в этом году.

```
sqlite> select * from products
   ...> where store_date like '2022%';
id          name        price       days_left   sort        date        store_date
----------  ----------  ----------  ----------  ----------  ----------  ----------
1           fish        55          4           red         1648242413  2022-03-29
3           meat        70          5           beef        2022-02-15  2022-02-20
4           popit       4           100         colorful    2021-11-14  2022-01-20
6           silk        199         150         chineese    2020-07-24  2022-03-10
1           fish        55          4           red         1648242413  2022-03-29
3           meat        70          5           beef        2022-02-15  2022-02-20
4           popit       4           100         colorful    2021-11-14  2022-01-20
6           silk        199         150         chineese    2020-07-24  2022-03-10
sqlite>
```

Используя инструкцию alter, добавить дополнительные столбцы, один из которых category_id (тип integer и содержит идентификаторы категорий товаров).

```
sqlite> alter table products add column category_id integer
   ...> ;
```

Создать таблицу category (id, cat_name, cat_description)

```
sqlite> create table category (
   ...> id int primary key not null,
   ...> cat_name text not null,
   ...> cat_description text not null);
sqlite> .tables
category  products
sqlite>
```

Вывести данные обо всех товарах в форме идентификатор товара, наименование, дата выпуска, название категории товара.

```
sqlite> select category.cat_name, name, date from products join category on products.category_id = category.id group by name;
cat_name    name        date
----------  ----------  ----------
food        bread       2022-04-10
tech        cellphone   2022-07-06
food        fish        2022-03-15
entertaime  popit       2021-11-16
materials   silk        2021-08-08
sqlite>
```

подсчет количества товаров с помощью count, если стоимость=49 руб

```
sqlite> select count(*) from products
   ...> where price = 49;
count(*)
----------
1
sqlite>
```

суммарная стоимость товаров с помощью sum, если год выпуска>2016

```
sqlite> select sum(price) from products
   ...> where date > '2016-01-01';
sum(price)
----------
1902
sqlite>
```

максимальная и минимальная стоимость с помощью max и min

```
sqlite> select max(price) from products;
max(price)
----------
99
sqlite>
```

```
sqlite> select min(price) from products;
min(price)
----------
1500
sqlite>
```

Используя инструкцию inner join вывести полные сведения о товарах и категории для категории с id=2.

```
sqlite> select category.id, category.cat_name, category.cat_description, products.id, name, price, days_left, sort, date, store_date  from products inner join category on category.id = 2 and products.cat
egory_id = 2 group by name;
id          cat_name     cat_description             id          name        price       days_left   sort        date        store_date
----------  -----------  --------------------------  ----------  ----------  ----------  ----------  ----------  ----------  ----------
2           entertaiment  products used for simple fun  3           popit       99          100         white       2021-11-16  2022-02-10
sqlite>
```

Задание 3. Управление базой данных в SQLite Database Manager

Упражнение 3.2

1. Составьте запрос к таблицам Categories и Spendings, который возвращает название категории покупки, название магазина и потраченную сумму для всех покупок, относящихся к категории с номером 2.

```
1  select Categories.Category, Shop, sum(Amount) from  Spendings inner join Categories on Spendings.Category_ID = 2 and Categories.Category_ID = 2;
```

	Category	Shop	sum(Amount)
1	Одежда	Camel Active	4940

2. Составьте запрос, возвращающий названия категорий с номерами 1, 3 и 4 (с использованием ключевого слова IN)

```
1  select * from Categories where Category_ID in (1, 3, 4);
```

	Category_ID	Category	Category_Description
1	1	Еда	Продукты и напитки (кроме ресторанов...
2	3	Гаджеты	Телефоны, планшеты, часы - всё, без че...
3	4	Развлечения	Кино, театры и прочие культурные ...

Упражнение 3.3.

Создать БД и выполнить запросы по вариантам Создать БД согласно варианту для задания 2 (упражнение 2.3). Продемонстрировать выполнение всех операций и запросов согласно упражнениям 2.3 и 2.4.

Упражнение 2.4

Вывести данные про товары срок годности которых истекает в этом году

```
select * from products where store_date like '2022%'
```

id	name	price	store	sort	date	store_date
1	fish	55	9	red	2022-02-02	2022-02-11
2	bread	49	5	white	2022-03-05	2022-03-10
3	popit	150	150	simple	2021-03-19	2022-01-15

Используя инструкцию alter, добавить дополнительные столбцы, один из которых category_id (тип integer и содержит идентификаторы категорий товаров).

id	name	price	store	sort	date	store_date	category_id
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	fish	55	9	red	2022-02-02	2022-02-11	NULL
2	bread	49	5	white	2022-03-05	2022-03-10	NULL
3	popit	150	150	simple	2021-03-19	2022-01-15	NULL
4	silk	299	500	cheap	2021-05-29	2023-01-01	NULL
5	cellphone	1500	2000	xiaomi	2020-10-10	2025-01-05	NULL

Создать таблицу category (id, cat_name, cat_description).

Table

category

Advanced

Fields

Add field | Remove field | Move field up | Move field down

Name	Type	NN	PK	AI	U	Default
id	INTEGER	✓	✓	✓	✓	
cat_name	TEXT	✓				
cat_description	TEXT	✓				

```
CREATE TABLE "category" (
    "id"    INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    "cat_name"  TEXT NOT NULL,
    "cat_description"   TEXT NOT NULL
);
```

Вывести данные обо всех товарах в форме идентификатор товара, наименование, дата выпуска, название категории товара.

```
select category.cat_name, name, date from products join category on products.category_id = category.id;
```

cat_name	name	date
food	fish	2022-02-02
food	bread	2022-03-05
entairtainment	popit	2021-03-19
materials	silk	2021-05-29
tech	cellphone	2020-10-10

подсчет количества товаров с помощью count, если стоимость=49 руб

```
select count(*) from products where price = 49;
```

count(*)
1

суммарная стоимость товаров с помощью sum, если год выпуска>2016

```
select sum(price) from products where date > 2016-01-01;
```

	sum(price)
1	2053

(P.s. цены немного изменились со времен второго задания)

максимальная и минимальная стоимость с помощью max и min

```
select max(price) from products;
```

max(price)
1500

```
select min(price) from products;
```

min(price)
49

Используя инструкцию inner join вывести полные сведения о товарах и категории для категории с id=2.

SQL 1

```
select category.id, category.cat_name, category.cat_description, products.id, name, price, store, sort, date, store_date from products inner join category on categor
```

	id	cat_name	cat_description	id	name	price	store	sort	date	store_date
1	2	entairtainment	just for fun	3	popit	150	150	simple	2021-03-19	2022-01-15

Упражнение 2.3

создание таблицы (create);

Table

products

Advanced

Fields

Add field | Remove field | Move field up | Move field down

Name	Type	NN	PK	AI	U	Default
price	INTEGER	✓				
store	INTEGER	✓				
sort	TEXT	✓				
date	INTEGER	✓				
store_date	INTEGER	✓				

```
CREATE TABLE "products" (
    "id"    INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
    "name"  TEXT NOT NULL,
    "price" INTEGER NOT NULL,
    "store" INTEGER NOT NULL,
    "sort"  TEXT NOT NULL,
    "date"  INTEGER NOT NULL,
    "store_date"    INTEGER NOT NULL
);
```

Cancel | OK

вставка данных в таблицу (insert);

id	name	price	store	sort	date	store_date
Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	fish	55	9	red	2022-02-02	2022-02-11
2	bread	49	5	white	2022-03-05	2022-03-10
3	popit	150	150	simple	2021-03-19	2022-01-15
4	silk	299	500	cheap	2021-05-29	2023-01-01
5	cellphone	1500	2000	xiaomi	2020-10-10	2025-01-05

выборка данных (select) с выводом всех данных по столбцам и строкам, с сортировкой по id и по имени и с выводом последних 5 строк (инструкция limit);

```
select * from products order by id desc limit 5;
```

id	name	price	store	sort	date	store_date	category_id
5	cellphone	1500	2000	xiaomi	2020-10-10	2025-01-05	4
4	silk	299	500	cheap	2021-05-29	2023-01-01	3
3	popit	150	150	simple	2021-03-19	2022-01-15	2
2	bread	49	5	white	2022-03-05	2022-03-10	1
1	fish	55	9	red	2022-02-02	2022-02-11	1

выборка данных с фильтрацией (условие where), если id=5;

```
select * from products where id = 5;
```

id	name	price	store	sort	date	store_date	category_id
5	cellphone	1500	2000	xiaomi	2020-10-10	2025-01-05	4

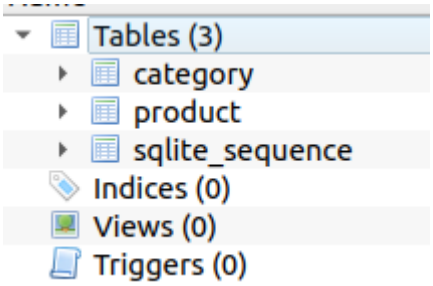
выборка данных с фильтрацией (условие where) и с совпадением по маске, например все записи, где имя объекта (согласно варианту) начинается на первую букву вашей фамилии (инструкция like);

```
select * from products where name like 'c%';
```

id	name	price	store	sort	date	store_date	category_id
5	cellphone	1500	2000	xiaomi	2020-10-10	2025-01-05	4

переименование таблицы (alter);

```
alter table products rename to product;
```



обновление данных с использованием update;

```
update product set price = 1499 where name = 'cellphone';
```

5	cellphone	1499	2000	xiaomi	2020-10-10	2025-01-05	4

удаление строк по id и по названию объекта;

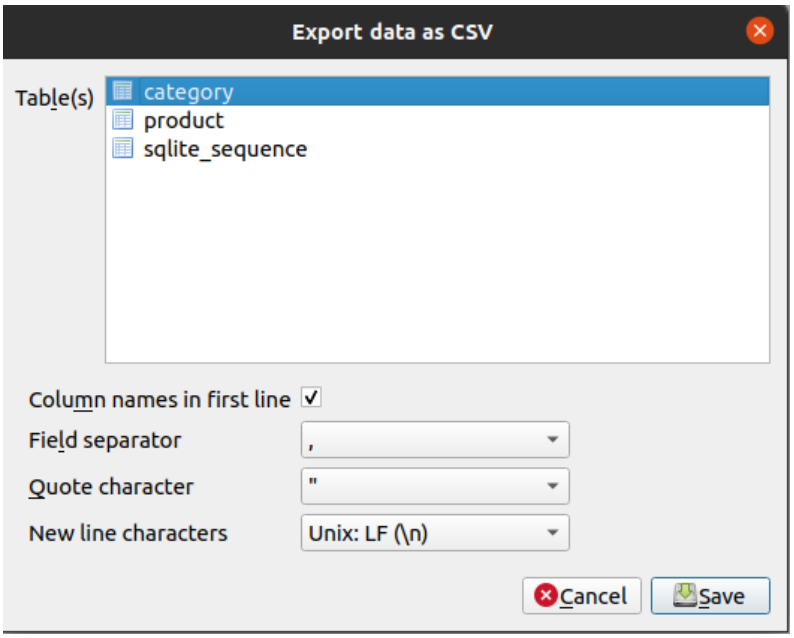
```
delete from product where id = 2;
```

1	fish	55	9	red	2022-02-02	2022-02-11	1
3	popit	150	150	simple	2021-03-19	2022-01-15	2
4	silk	299	500	cheap	2021-05-29	2023-01-01	3
5	cellphone	1499	2000	xiaomi	2020-10-10	2025-01-05	4

```
delete from product where name = 'cellphone';
```

1	fish	55	9	red	2022-02-02	2022-02-11	1
3	popit	150	150	simple	2021-03-19	2022-01-15	2
4	silk	299	500	cheap	2021-05-29	2023-01-01	3

экспорт базы данных в файлы .sql, .csv.;



Задание 4. Изучение примеров приложений на C подключения и запросов к базе данных

Репозиторий: https://github.com/fpmi-tp2022/labrabota4-gr15-alekseykrazhev

Пример 1:

```
alex@AVOCADOBOOK:~/Desktop/study/4 sem/TP/lab4/labrabota4-gr15-alekseykrazhev/ex
amples$ gcc example1_gr15_KrazhevskiyAleksey.c -l sqlite3
alex@AVOCADOBOOK:~/Desktop/study/4 sem/TP/lab4/labrabota4-gr15-alekseykrazhev/ex
amples$ ./a.out
Opened database successfully
alex@AVOCADOBOOK:~/Desktop/study/4 sem/TP/lab4/labrabota4-gr15-alekseykrazhev/ex
amples$
```

Пример 2:

```
alex@AVOCADOBOOK:~/Desktop/study/4 sem/TP/lab4/labrabota4-gr15-alekseykrazhev/ex
amples$ gcc example2_gr15_KrazhevskiyAleksey.c -l sqlite3
alex@AVOCADOBOOK:~/Desktop/study/4 sem/TP/lab4/labrabota4-gr15-alekseykrazhev/ex
amples$ ./a.out
Opened database successfully
Table created successfully
alex@AVOCADOBOOK:~/Desktop/study/4 sem/TP/lab4/labrabota4-gr15-alekseykrazhev/ex
amples$
```

Пример 3:

```
alex@AVOCADOBOOK:~/Desktop/study/4 sem/TP/lab4/labrabota4-gr15-alekseykrazhev/ex
amples$ gcc example3_gr15_KrazhevskiyAleksey.c -l sqlite3
alex@AVOCADOBOOK:~/Desktop/study/4 sem/TP/lab4/labrabota4-gr15-alekseykrazhev/ex
amples$ ./a.out
Opened database successfully
Records created successfully
alex@AVOCADOBOOK:~/Desktop/study/4 sem/TP/lab4/labrabota4-gr15-alekseykrazhev/ex
amples$
```

Пример 4:

```
alex@AVOCADOBOOK:~/Desktop/study/4 sem/TP/lab4/labrabota4-
amples$ gcc example4_gr15_KrazhevskiyAleksey.c -l sqlite3
alex@AVOCADOBOOK:~/Desktop/study/4 sem/TP/lab4/labrabota4-
amples$ ./a.out
Opened database successfully
Callback function called: ID = 1
NAME = Paul
AGE = 32
ADDRESS = California
SALARY = 20000.0

Callback function called: ID = 2
NAME = Allen
AGE = 25
ADDRESS = Texas
SALARY = 15000.0

Callback function called: ID = 3
NAME = Teddy
AGE = 23
ADDRESS = Norway
SALARY = 20000.0

Callback function called: ID = 4
NAME = Mark
AGE = 25
ADDRESS = Rich-Mond
SALARY = 65000.0

Operation done successfully
```

Пример 5:

```
alex@AVOCADOBOOK:~/Desktop/study/4 sem/TP/lab4/labrabota4-
amples$ gcc example5_gr15_KrazhevskiyAleksey.c -l sqlite3
alex@AVOCADOBOOK:~/Desktop/study/4 sem/TP/lab4/labrabota4-
amples$ ./a.out
Opened database successfully
Callback function called: ID = 1
NAME = Paul
AGE = 32
ADDRESS = California
SALARY = 25000.0

Callback function called: ID = 2
NAME = Allen
AGE = 25
ADDRESS = Texas
SALARY = 15000.0

Callback function called: ID = 3
NAME = Teddy
AGE = 23
ADDRESS = Norway
SALARY = 20000.0

Callback function called: ID = 4
NAME = Mark
AGE = 25
ADDRESS = Rich-Mond
SALARY = 65000.0

Operation done successfully
```

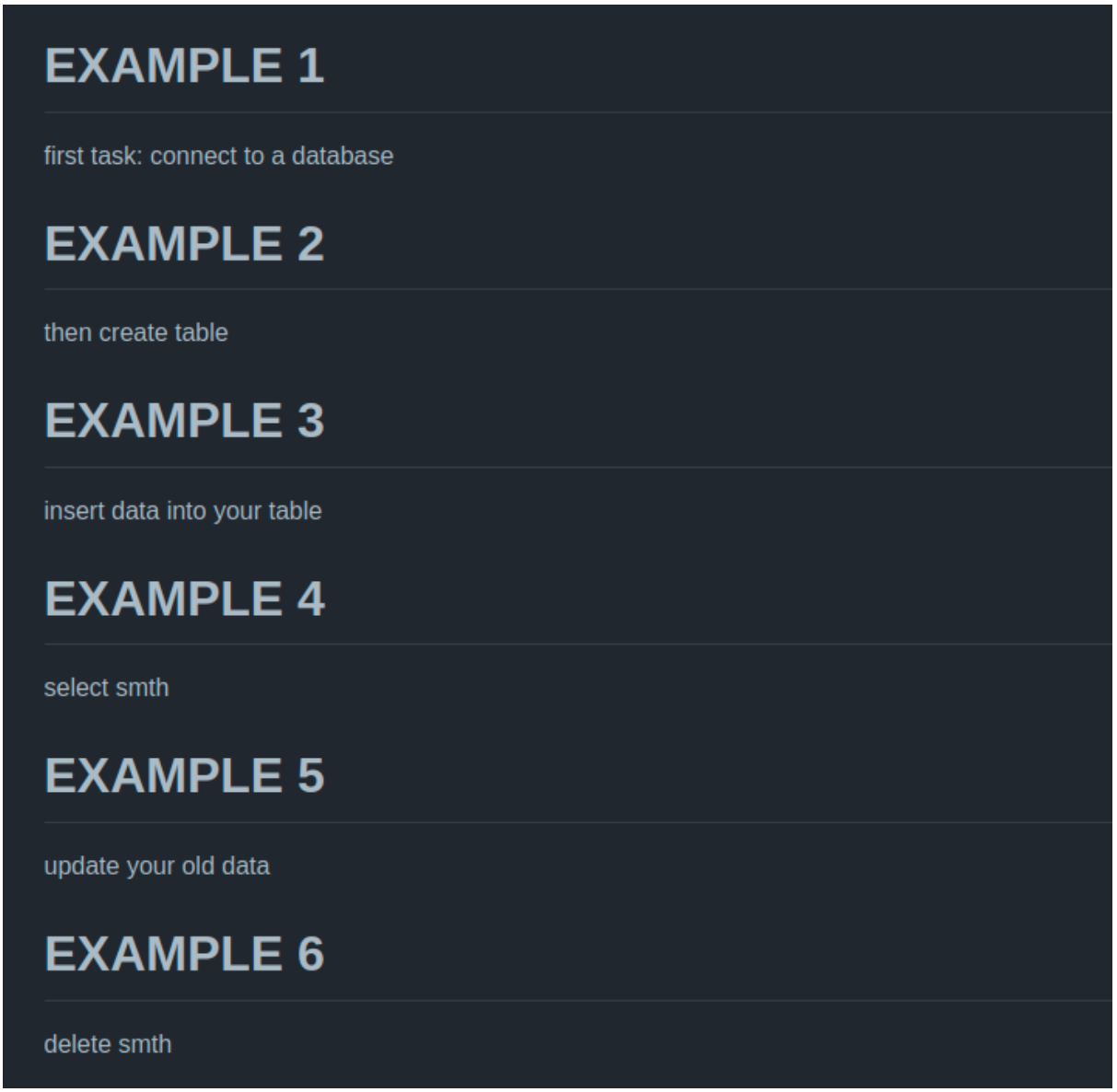
Пример 6:

```
alex@AVOCADOBOOK:~/Desktop/study/4 sem/TP/lab4/labrabota
amples$ gcc example6_gr15_KrazhevskiyAleksey.c -l sqlite
alex@AVOCADOBOOK:~/Desktop/study/4 sem/TP/lab4/labrabota
amples$ ./a.out
Opened database successfully
Callback function called: ID = 1
NAME = Paul
AGE = 32
ADDRESS = California
SALARY = 25000.0

Callback function called: ID = 3
NAME = Teddy
AGE = 23
ADDRESS = Norway
SALARY = 20000.0

Callback function called: ID = 4
NAME = Mark
AGE = 25
ADDRESS = Rich-Mond
SALARY = 65000.0

Operation done successfully
```

Пример 6 (код):

```c
#include <stdio.h>
#include <stdlib.h>
#include <sqlite3.h>

static int callback(void *data, int argc, char **argv, char **azColName) {
   int i;
   fprintf(stderr, "%s: ", (const char*)data);

   for(i = 0; i<argc; i++) {
      printf("%s = %s\n", azColName[i], argv[i] ? argv[i] : "NULL");
   }
   printf("\n");
   return 0;
}

int main(int argc, char* argv[]) {
   sqlite3 *db;
   char *zErrMsg = 0;
   int rc;
   char *sql;
   const char* data = "Callback function called";

   /* Open database */
   rc = sqlite3_open("test.db", &db);

   if( rc ) {
      fprintf(stderr, "Can't open database: %s\n", sqlite3_errmsg(db));
      return(0);
   } else {
      fprintf(stderr, "Opened database successfully\n");
   }

   /* Create merged SQL statement */
   sql = "DELETE from COMPANY where ID=2; " \
         "SELECT * from COMPANY";

   /* Execute SQL statement */
   rc = sqlite3_exec(db, sql, callback, (void*)data, &zErrMsg);

   if( rc != SQLITE_OK ) {
      fprintf(stderr, "SQL error: %s\n", zErrMsg);
      sqlite3_free(zErrMsg);
   } else {
      fprintf(stdout, "Operation done successfully\n");
   }
   sqlite3_close(db);
   return 0;
}
```

EXAMPLES.md:



README.md:

README.md

OWERVIEW

alekseykrazhev REPO for lab4 TP all branches: main, examples examples branch contains c-code examples of working with SQLite3

USAGE

task 4 solution

BUILDING

gcc .c -l sqlite3

ADDITIONAL NOTES

Report: https://docs.google.com/document/d/1WJzLoEc6EV1C94y_dDgDS6BEwSYb5yyGE-VqwoCUESQ/edit

Задание 5. Создание приложения на C для подключения к базе данных и запросы к базе данных

Создана новая ветка dev:

```
project5$ git branch
* dev
  examples
  master
```

Makefile:

```
task 5: obj/main.o
	gcc -o bin/task5 obj/main.o -l sqlite3

obj/main.o: src/main.c
	gcc -o obj/main.o -c src/main.c

clean:
	rm -rf obj/*.o bin/task5
~
```

Сборка проекта:

```
alex@AVOCADOBOOK:~/Desktop/study/4 sem/TP
Opened database successfully
Callback function called: id = 1
name = fish
price = 55
store = 9
sort = red
date = 2022-02-02
store_date = 2022-02-11
category_id = 1

Callback function called: id = 3
name = popit
price = 150
store = 150
sort = simple
date = 2021-03-19
store_date = 2022-01-15
category_id = 2

Callback function called: id = 4
name = silk
price = 299
store = 500
sort = cheap
date = 2021-05-29
store_date = 2023-01-01
category_id = 3

Callback function called: id = 5
name = cellphone
price = 1500
store = 2000
sort = xiaomi
date = 2021-02-05
store_date = 2025-10-10
category_id = 4

Callback function called: id = 6
name = bread
price = 49
store = 5
sort = white
date = 2022-03-05
store_date = 2022-03-10
category_id = 1
```

Меню выбора:

```
Opened database successfully
1 - SELECT operation, 2 - INSERT operation, 3 - DELETE operation, 4 - request, 5 - exit
```

Select (1) вывести таблицу полностью:

```
Opened database successfully
1 - SELECT operation, 2 - INSERT operation, 3 - DELETE operation, 4 - request, 5 - exit
1
Callback function called: id = 1
name = fish
price = 55
store = 9
sort = red
date = 2022-02-02
store_date = 2022-02-11
category_id = 1

Callback function called: id = 3
name = popit
price = 150
store = 150
sort = simple
date = 2021-03-19
store_date = 2022-01-15
category_id = 2

Callback function called: id = 4
name = silk
price = 299
store = 500
sort = cheap
date = 2021-05-29
store_date = 2023-01-01
category_id = 3

Callback function called: id = 5
name = cellphone
price = 1500
store = 2000
sort = xiaomi
date = 2021-02-05
store_date = 2025-10-10
category_id = 4

Callback function called: id = 6
name = bread
price = 49
store = 5
sort = white
date = 2022-03-05
store_date = 2022-03-10
category_id = 1

Operation done successfully
```

Insert (2) добавить запись:

```
Opened database successfully
1 - SELECT operation, 2 - INSERT operation, 3 - DELETE operation, 4 - request, 5 - exit
2
Enter id
2
Enter name
vine
Enter price
89
Enter store
500
Enter sort
red
Enter date
2017-05-05
Enter store_date
2021-05-05
Enter category_id
1
INSERT INTO product (id, name, price, store, sort, date, store_date, category_id) VALUES (2, 'vine', 89, 500, 'red', strftime('%Y-%m-%d', '2017-05-05'), strftime('%Y-%m-%d', '2021-05-05'), 1);
Operation done successfully
```

Delete (3) удалить запись (по id или имени (его части)):

```
alex@AVOCADOBOOK:~/Desktop/study/4 sem/TP/lab4/labrabota4-gr15-alekseykrazhev/bd
project5$ ./bin/task5
Opened database successfully
1 - SELECT operation, 2 - INSERT operation, 3 - DELETE operation, 4 - request, 5
 - exit
3
1 - id, 2 - name
2
Enter name of product fi
Operation done successfully
alex@AVOCADOBOOK:~/Desktop/study/4 sem/TP/lab4/labrabota4-gr15-alekseykrazhev/bd
project5$
```

```
Opened database successfully
1 - SELECT operation, 2 - INSERT operation, 3 - DELETE operation, 4 - request, 5 - exit
3
Enter ID of product 2
Operation done successfully
```

Request (4) запрос вывода с определенного id или имени (части) или по общему для нескольких строк (category_id):

```
Opened database successfully
1 - SELECT operation, 2 - INSERT operation, 3 - DELETE operation, 4 - request, 5
 - exit
4
1 - id, 2 - name
2
Enter name of product fi
Callback function called: id = 1
name = fish
price = 55
store = 9
sort = red
date = 2022-02-02
store_date = 2022-02-11
category_id = 1

Operation done successfully
```

```
alex@AVOCADOBOOK:~/Desktop/study/4 sem/TP/lab4/labrabota4-gr15-alekseykrazhev/bdproject5$ ./bin/task5
Opened database successfully
1 - SELECT operation, 2 - INSERT operation, 3 - DELETE operation, 4 - request, 5 - exit
4
Enter ID of product 3
Callback function called: id = 3
name = popit
price = 150
store = 150
sort = simple
date = 2021-03-19
store_date = 2022-01-15
category_id = 2

Operation done successfully
```

```
Opened database successfully
1 - SELECT operation, 2 - INSERT operation, 3 - DELETE operation, 4 - request, 5
 - exit
4
1 - id, 2 - name, 3 - category_id
3
Enter category_id of products 1
Callback function called: id = 6
name = bread
price = 49
store = 5
sort = white
date = 2022-03-05
store_date = 2022-03-10
category_id = 1

Callback function called: id = 7
name = milk
price = 45
store = 7
sort = new
date = 2022-05-05
store_date = 2022-05-12
category_id = 1

Operation done successfully
```

Exit (5):

```
alex@AVOCADOBOOK:~/Desktop/study/4 sem/TP/lab4/labrabota4-gr15-alekseykrazhev/bdproject5$ ./bin/task5
Opened database successfully
1 - SELECT operation, 2 - INSERT operation, 3 - DELETE operation, 4 - request, 5 - exit
5
Goodbye!
alex@AVOCADOBOOK:~/Desktop/study/4 sem/TP/lab4/labrabota4-gr15-alekseykrazhev/bdproject5$
```

Функционал для вставки данных в режиме autocommit (исполняется sqlite3 построчно) и транзакцией:

```
const char *transaction = "BEGIN TRANSACTION;select * from products; insert into
products values (11, 'name', price, store, 'sort', 'date', 'store_date',
category_id); COMMIT;";
```

Контрольные вопросы:

1. Перечислите способы создания базы данных sqlite.

Через соответствующую программу (BD Browser, Valentina Studio), через терминал (командой create table). Также можно использовать импорт БД из файла.

2. С помощью какой команды в консоли sqlite можем просмотреть список баз данных и подключённых файлов баз данных?
.databases

3. Приведите перечень команд для экспорта данных из таблицы базы данных в файл с расширением .csv.

.headers on
.mode csv
.output <filename>
.dump.output stdout

4. Приведите перечень команд для экспорта отдельной таблицы и всей базы данных в файл с расширением .sql и сжатый файл, например в файл с расширением .sql.tgz.

.headers on
.mode csv
.output <filename>.sql

5. Как вывести из таблицы данных по строкам и по столбцам?
Команда .mode columns
select <что-то> <откуда>

6. Для чего используется команда .headers в консоли sqlite?
Эта команда включает заголовки в таблице

7. Какая команды используется для вывода настроек окружения в sqlite?
.show

8. С помощью какой команды выводится список таблиц базы данных в консоли sqlite?
.tables

9. Приведите пример запроса выборки из 2-х таблиц.
select * from products inner join category on products.cat_id = category.id

10.Приведите пример запроса для обновления данных в строках таблицы в зависимости от значения определенного поля.
update products set price = 150 where name = ‘fish’

11. Приведите пример функции, которая открывает соединение с файлом базы данных SQLite и возвращает объект соединения с базой данных, который будет использоваться другими функциями SQLite?
rc = sqlite3_open(“products”, &db);

12.Приведите пример синтаксиса функции sqlite3_exec и объясните результат выполнения.

rc = sqlite3_exec(db, sql, callback, 0, &zErrMsg);

При возникновении ошибки эта команда вернет ее описание (код). При правильном выполнении возвращается SQLITE_OK

13.Какая функция закрывает соединение с базой данных, ранее открытое вызовом sqlite3_open()? Приведите пример синтаксиса
sqlite3_close(db);

14.Приведите пример фрагмента кода на языке C для создания таблицы в базе данных sqlite и объясните его

```
/* Create SQL statement */
sql = "CREATE TABLE COMPANY("  \
   "ID INT PRIMARY KEY     NOT NULL," \
   "NAME           TEXT    NOT NULL," \
   "AGE            INT     NOT NULL," \
   "ADDRESS        CHAR(50)," \
   "SALARY         REAL );";

/* Execute SQL statement */
rc = sqlite3_exec(db, sql, callback, 0, &zErrMsg);
```

Мы записываем запрос для создания нужной таблицы в строке sql, а затем передаем его на выполнение в БД командой sqlite3_exec.

15.Приведите пример фрагмента кода на языке C для вставки данных в таблицу в базе данных sqlite и объясните его.

```
/* Create SQL statement */
sql = "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) "  \
      "VALUES (1, 'Paul', 32, 'California', 20000.00 ); " \
      "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY) "  \
      "VALUES (2, 'Allen', 25, 'Texas', 15000.00 ); "     \
      "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)" \
      "VALUES (3, 'Teddy', 23, 'Norway', 20000.00 );" \
      "INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)" \
      "VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00 );";

/* Execute SQL statement */
rc = sqlite3_exec(db, sql, callback, 0, &zErrMsg);
```

Создаем строку sql, в которую записываем нужный запрос, а затем передаем этот запрос на выполнение в БД командой sqlite3_exec

16.Приведите пример фрагмента кода на языке C выполнением AUTOCOMMIT и TRANSACTION и объясните в чем особенности использования их.

```
const char *transaction = "BEGIN TRANSACTION;select * from products; insert into
products values (11, 'name', price, store, 'sort', 'date', 'store_date',
category_id); COMMIT;";
```

Используя AUTOCOMMIT каждая команда выполняется по очереди, друг за другом. Следовательно, при возникновении каких-либо ошибок прервать и сделать откат не получится
А при использовании TRANSACTION все команды будут выполняться сразу, и значит сделать откат при необходимости возможно.